# Exhibit 14

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

| | | |
|---|---|---|
| ORACLE AMERICA, INC., | ) | |
| | ) | |
| Plaintiff, | ) | |
| | ) | |
| v. | ) | Civ. A. No. 10-03561 WHA |
| | ) | |
| GOOGLE INC., | ) | *(Jury)* |
| | ) | |
| Defendant. | ) | |

**EXPERT REPORT OF PROFESSOR DOUGLAS C. SCHMIDT, Ph.D.**

**January 8, 2016**

**Highly Confidential – Attorneys' Eyes Only**

cross-platform benefits with respect to portability and underlying technology architecture.[4] Both Sun and Oracle have actively sought to preserve compatibility.[5]

26. Google's copying of the 37 Java APIs is a reflection of the value of the Java API packages' key attributes, including benefits to developer productivity (e.g., leveraging the large pool of developers familiar with the Java API packages) and benefits related to the creative expression reflected in the packages, as well as avoiding the technical downsides of alternative platforms available during the mid-2000s.

27. Google's use of the 37 Java APIs in the Android platform violates the fundamental "Write Once, Run Anywhere" philosophy of the Java ecosystem. In addition to failing to meet the expectations of compatibility that Oracle has fostered in the community, Google's use of the 37 Java APIs in Android represents both a subset and superset of the Java SE API Specification and thus fails to demonstrate compatibility with Java when subjected to various forms of compatibility testing, including Oracle's industry benchmark Technology Compatibility Kit ("TCK," or sometimes referred to as or used interchangeably with the Java Compatibility Kit or "JCK"), as well as independent tests and measurements. Moreover, Google's incompatible[6] use of the 37 Java APIs in Android can be seen in both Android's software development kits ("SDKs") and Android's virtual machine runtime environments.

28. The Android platform is critically dependent on the 37 Java API packages at issue, individually and collectively, and the copied declaring code. I demonstrate this critical dependence by showing that without the 37 Java API packages at issue, individually and collectively, and the copied declaring code, the build process used to create the code that runs Android on mobile devices fails to generate the necessary executable code (called executable

[4] Lindholm, T., Yellin, F., Bracha, G., & Buckley, A. (2014). *The Java Virtual Machine Specification*. Pearson Education., p. 1 ("The java platform was initially developed to address the problems of building software for networked consumer devices. It was designed to support multiple host architectures and to allow secure delivery of software components. To meet these requirements, compiled code had to survive transport across networks, operate on any client, and assure the client that it was safe to run.") http://tech-insider.org/java/research/1996/0123.html

[5] The compatibility guide for Java 8 is described at http://www.oracle.com/technetwork/java/javase/8-compatibility-guide-2156366.html.

[6] When I describe "incompatible" uses in this report, unless otherwise noted, I mean the incompatibility between Google's use of the 37 API packages and the suite of API packages available in various editions of Java SE (JDK). When I write about "compatibility" in this report, unless otherwise noted, I mean conformance with the Java Compatibility Kit (JCK).

image files). The build process fails if files from even one of the 37 Java API packages at issue is removed. In fact, the build process also fails when I remove only the copied lines of declaring code while leaving the rest of the files in place. As a result, Android will not compile, let alone run on a mobile device without all of the copied declaring code and the full set of files for the 37 Java API packages. In short, without any or all of 37 Java API packages at issues and the copied declaring code, mobile devices running Google's Android are unusable.

## V. BACKGROUND ON THE JAVA AND ANDROID PLATFORMS

### A. Background on the Java Platform

29. Sun Microsystems first released the Java programming platform in 1995. In 2010, Oracle Corporation (also "Oracle") acquired Sun. The Java platform contains three distinct resources: the Java Language, the Java Virtual Machine, and the Java API packages (sometimes referred to as the Java Class Libraries).[7] Java is one of the most popular programming platforms in the world.[8] The Java platform, including the Java API packages at issue in this case, evolved through multiple iterations, as is shown in Appendix D – Evolution of Java APIs. The Java API packages enable and accelerate the development of a wide range of different programs and applications across a wide range of operating systems and hardware platforms. In this section, I provide a description of the Java Platform and Java API packages. I provide a glossary of terms relevant to this section in Appendix E - Glossary of Terms.

30. The Java platform, and particularly the Java API packages, enables a large and active community of software developers. Those software developers create computer programs (also known as applications or simply apps). The programs run on computing devices like personal computers and mobile phones. The group of software developers who write programs for a given platform are known as that platform's community (e.g., the Java community).

[7] A representation of the Java platform is at Appendix C - The Java Class Libraries. *See* JDK 5.0 Documentation, Sun, https://web.archive.org/web/20100330080522/http://java.sun.com/j2se/1.5.0/docs/index.html (last visited January 7, 2016).

[8] Stephen Cass, *The 2015 Top Ten Programming Languages,* IEEE Spectrum (July 20, 2015) http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages.

## VIII. WITHOUT THE COPIED ORACLE DECLARING CODE AND API PACKAGES, ANDROID WILL NOT COMPILE AND IS RENDERED INOPERABLE

78. Google copied the 37 Java APIs in a manner that renders the operation of the Android platform critically dependent on each API package at issue. I find that Android is not usable on a computing device, such as a phone or tablet, without each of the Java APIs packages at issue or the copied declaring code in them. Regardless of the size of the Java API code lines in the operating system profile, their removal is fatal to Android's operability. Just as the brain stem, heart or pancreas represent fractional shares of a human being's overall body mass, their surgical removal would clearly be of far greater consequence than the amputation of an entire limb.

### A. Background on the Android build process

79. The Android build process broadly refers to the compiling and assembling of the source code files, such that they are ready to run on an actual Android device. The build is the creation of the actual Android platform from the source code.

80. After a build is completed, system image files are produced. As I explained earlier, machine code is code that can be executed. When I talk about the system image file being produced, I am talking about the source code for the Android platform being compiled into machine code. System image files are then 'flashed' onto a device. Flashing refers to the process by which the code of the Android platform is stored on the device. If the device successfully boots a working version of Android from these images, it means the build was successful. An image that fails to boot on the device, or a build that does not produce these images, is treated as a failed build.[33]

---

combinations. In fact, this number ends up looking very small when you factor in the extra dimension of function objects that can customize most of the algorithms."

*See:* Laurence Vanhelsuwé, *Need a good set of abstract data structures? ObjectSpace's JGL packs a punch!,* JavaWorld (Jan. 1, 1997), http://www.javaworld.com/article/2076958/java-app-dev/need-a-good-set-of-abstract-data-structures--objectspace-s-jgl-packs-a-punch-.html.

[33] The system.img, userdata.img, and boot.img files are sufficient to launch Android when flashed onto a device, and the system.img image file is necessary to start Android on a device. As a result, these are the three image files that will be searched for once a build test has completed.

made in that particular experiment.[35] No material from the source code other than the copied 37 API packages was modified in any way during any of these tests.

86. As a control case for the tests, a single successful build of the Android platform was performed using a server instance generated from this build environment. This control case verified that the environment used for the tests produces a successful build when the source code is left unaltered.

### C. Results

87. I find that the 37 Java API packages at issues and their copied declaring code are required for the build process to complete successfully. Accordingly, they are required for the functioning of the Android platform.

88. I performed three different types of tests to understand and corroborate the robustness and breadth of my findings. I describe the methodology and the results of each test in detail in this sub-section.

#### 1) Android will not function when all copied 37 Java API packages at issue are removed

89. The first build test involved attempting the build process after removing all source files belonging to the copied 37 Java API packages at issue from the Android source code. The actual directories in which the source files are located were not deleted - only the source files inside them were removed. Once this material was removed, the build was attempted to see if it was able to complete successfully without the material from the 37 Java API packages at issue. During this attempt, the build failed in the following way – the build returned an error and aborted. Appendix K – Error Log from Android Build Test No. 1: Removal of the Entire Set of 37 Copied APIs shows the full error log for this build attempt.

90. Verifying the results through the record of the build contained in the 'out' folder revealed that building without the copied 37 Java API packages at issue failed to produce the image files that are required to run Android on a device. This result indicates that Android fails to build

[35] The source files from the 37 Java API packages at issue that are modified in this experiment are found in the libcore/luni/src/main/java folder of the source code root directory.

successfully when the 37 Java API packages at issue are removed. This failure to build illustrates that Android is unusable without the presence of the 37 Java API packages at issue.

2) Android will not function if any one of the copied 37 Java API packages at issue is removed

91. The second set of build tests attempted the Android build when each of the copied 37 Java API packages at issue was individually removed from the source code. This test was repeated 37 different times, each time removing a different copied API package. For every test attempted, the build aborted after returning an error. Appendix L - Error Log from Android Build Test No. 2: Removal of Individual API Source Files, shows the error logs for the 37 repetitions.

92. Searching through the build output once again showed that the system images that are needed to boot Android on a device were not created; this same result was observed for all 37 iterations. Together, these results show that Android fails to build successfully when any single one of the 37 copied API packages at issue is removed. These 37 failures to build illustrate that Android is unusable in the absence of any one the 37 Java API packages at issue.

3) Android will not function if the copied lines of copied declaring code are deleted from the source files of the copied 37 Java API packages at issue

93. The third and final test measured the dependence of the Android build process on the individual lines of copied declaring code inside each of the 37 Java API packages at issue. To this end, the build was attempted after identifying and deleting these declaring lines from the source files of the 37 Java API packages at issue from the source code.

94. It is important to note that this process did not involve deleting any actual source files; only the specific lines of code from inside the source files that were classified as copied declaration lines were removed. This removal was accomplished using the results of a prior analysis by Mr. Zeidman that identifies which particular lines in which source files constitute copied declarations. In particular, I have reviewed the expert report of Mr. Zeidman including Exhibit S of that report in which he lists the declaring code copied by Google. It is this copied declaring code that I removed from the source code files of the 37 packages in order to conduct my tests. Android will not function if the copied lines of declaring code are deleted from the source file of any copied class.

a) Android will not function if the copied lines of declaring code are deleted from the source files of a single class of the copied 37 Java API packages at issue

95. As a first test, I took all of the copied declaring code out of the class java.text.annotation while leaving all other code intact. I then attempted to build the Android platform and it failed. When the build was attempted without each of the copied declarations, the build process returned an error and aborted prematurely every time. The complete error log summarizing the output of this build attempt can be found in Appendix M - Error Log from Android Build Test No. 3A: Removal of Source File Declaring Codes from java.text.annotation.

b) Android will not function if the copied lines of declaring code are deleted from the source files of the copied 37 Java API packages at issue

96. Next, I performed a final test by removing the copied declaring codes from all of the copied 37 Java API packages at issue. Once again, the output of the build for this test did not contain the necessary system image files. The findings from this test demonstrates that Android fails to build successfully when the copied declaring code lines are specifically removed, while leaving all other non-copied lines of code intact. These failures to build illustrate that Android is inoperable in the absence of any one of the copied declaring codes from inside the copied Java APIs. The complete error log summarizing the output of this build attempt can be found in Appendix N - Error Log from Android Build Test No. 3B: Removal of all copied declaring codes from all 37 APIs.

## IX. THE ANDROID PLATFORM VIOLATES THE FUNDAMENTAL "WRITE ONCE, RUN ANYWHERE" PARADIGM OF THE JAVA PLATFORM

### A. The Android Platform is incompatible based on the Java Compatibility Kit

97. I understand from the declaration of Mark Reinhold, Chief Architect of the Java Platform Group at Oracle, that Google's implementation of the 37 Java APIs in Android has failed to pass the official compatibility tests which are based on the Java Specifications.

98. The Java Compatibility Kit (JCK) is Java's official test suite for determining compatibility. It is used by Oracle to confirm compatibility of alleged implementations of Java with the adopted